Volume: 10 Issue: 01 Pages: 3704-3712 (2018) ISSN: 0975-0290

What happens when adaptive video streaming players compete in time-varying bandwidth conditions?

Koffka Khan
Department of Computing and Information Technology The University of the West Indies, Trinidad and Tobago, W.I
Email: koffka.khan@gmail.com
Wayne Goodridge
Department of Computing and Information Technology The University of the West Indies, Trinidad and Tobago, W.I
Email: wayne.goodridge@sta.uwi.edu.com

ABSTRACT

Competition among adaptive video streaming players severely diminishes user-QoE. When players compete at a bottleneck link many do not obtain adequate resources. This imbalance eventually causes ill effects such as screen flickering and video stalling. There have been many attempts in recent years to overcome some of these problems. However, added to the competition at the bottleneck link there is also the possibility of varying network bandwidth which can make the situation even worse. This work focuses on such a situation. It evaluates current heuristic adaptive video players at a bottleneck link with time-varying bandwidth conditions. Experimental setup includes the TAPAS player and emulated network conditions. The results show PANDA outperforms FESTIVE, ELASTIC and the Conventional players.

Keywords— **Adaptive video streaming, bottleneck, flickering, stalling, time-varying, bandwidth, TAPAS, PANDA, FESTIVE, ELASTIC**

Date of Submission: June 10, 2018 Date of Acceptance: June 23, 2018

I. INTRODUCTION

The concept of adaptive video streaming is based on the idea to adapt the bandwidth required by the video stream to the throughput available on the network path from the stream source to the client [27]. These algorithms can live at the server [2], at an intermediate network device [28] or at the client [17]. With the client-side approach it is the player that decides what bitrate to request for any fragment, improving server-side scalability [2]. A benefit of this approach is that the player can control its playback buffer size by dynamically adjusting the rate at which new fragments are requested. The adaptation is performed by varying the quality of the streamed video. Multiple video segments constitute a video stream lasting from as little as 2 seconds to as much as having a 10 second chunk delivery rate. Segments are encoded and stored on the server in numerous quality versions, termed representations. Each version has a unique resolution, bitrate and/or quality. A client downloads segments using HTTP GET statements [26]. However, with adaptive streaming a client might request subsequent segments at different quality levels to manage varying network conditions, based on an estimation bandwidth. To do this it uses a manifest file that contains information about the video segments. Protocols and standards such as MPEG Dynamic Adaptive Streaming over HTTP (DASH), Apple HTTP Live Streaming (HLS), Microsoft Smooth Streaming (MSS) or Adobe HTTP Dynamic Streaming (HDS) typically use a media playlist that contains a list of uniform resource identifiers (URIs) that are addresses to media segments [36]. The process of determining the ideal representation for each segment to enhance the user's experience is pivotal to adaptive streaming. The controller algorithm estimates the network bandwidth and chooses the next bitrate level corresponding to the available network bandwidth. Variations in the available bandwidth will result in jerky playback and disruption of the video playback if the throughput falls below the bit rate requirement of the video. This is the major challenge in adaptive video streaming. Selecting appropriate bitrate levels helps to maximize the user experience. Generally, higher bitrates and resolutions will give better user experience. For example, if a client approximates that there is 9.5Mb/s available in the network, it might request the server to stream video compressed to the highest video rate available, 9.5Mb/s, or the next rate below, 9.3Mb/s. If the client picks a video rate that is too high, the viewer will experience annoying re-buffering events; if they pick a streaming rate that is too low, the viewer will experience poor video quality. In both cases, the experience degrades 0, [19], [40], and user may take their viewing elsewhere [10]. It is therefore important for a video streaming service to select the highest safe video rate. [13].

Adaptive streaming uses the HTTP/TCP protocol stack to transmit video Web traffic. Thus, the development of this wave of HTTP-based streaming applications is not referred to as adaptive streaming over HTTP. The use of HTTP/TCP protocols for video streaming is because of the advantages that HTTP/TCP offers. It allows standard web servers and caches to be used increasing its' cost effectiveness. Another advantage is that all firewalls are configured to support HTTP connections [27]. In addition,

Volume: 10 Issue: 01 Pages: 3704-3712 (2018) ISSN: 0975-0290

is allows better scaling as HTTP is stateless and the streaming session is managed by the client, thus reducing the load on the server. However, HTTP/TCP use reveals further challenges as adaptation is on top of TCPs congestion control algorithm, which forms nested control loops. As the throughput of the TCP connection depends on both the link capacity and the amount of congestion, the throughput can vary significantly over time [32].

Video over IP is becoming more and more important as we move further into the twenty-first century. The Internet is still growing rapidly, and more uses are being found for video users. These include real-time online visual assistance, video
learning, live event streaming, smart HDTVs, mobile phones, gaming devices, computers and visual communication among others. As the content quality is improving to meet end-user demands the bandwidth requirement for such devices is rapidly increasing. With increasing bandwidth demands and profuse video content, it is becoming likely that two or more adaptive streaming players may have to share a network bottleneck. This will result in a competition for available bandwidth. Example scenarios where this can take place are, when a number of people in the same household view similar or different videos simultaneously. Here, the domestic broadband access link is the shared bottleneck. Another instance of such competition is when many users watch the same live event (such as World Cup Soccer) online. The shared bottleneck may be an edge network link in this scenario. It has been previously observed that such competition can lead to performance issues. [3], [23], [29]
. The rest of paper is organized as follows. We give the problem definition related to this paper in Section 2. Section 3 explores different adaptive streaming approaches including PANDA, FESTIVE and ELASTIC. In Section 4, we look at time-varying bandwidth conditions exiting in computer communication networks. In section 5 we give the experimental setup. Section 6 gives the results. Finally, we give our conclusion in section 7.

II. PROBLEM DEFINITION

In the presence of competing HTTP-based adaptive streaming (HAS) clients the TCP throughput does not always faithfully represent the fair-share bandwidth [22]. Three performance issues that can take place when two or more adaptive streaming players share a network bottleneck and compete for available bandwidth are instability, unfairness and utilization [1]. It is shown that in the case of two competing video flows FESTIVE and ELASTIC provide a received video rate that oscillates around the fair share, but with an increased number of video level switches [8]. Depending on the temporal overlap of the ON-OFF [18] periods among competing players, they may not estimate their fair share correctly [1]. In the case where both players overestimate their fair share, they may request a video representation with a higher bitrate than the fair share, which causes network congestion. Consequently, the players measure that their TCP throughput is lower than their previous fair share estimate, and so switch down to a lower video bitrate level. This creates a repeating oscillatory scenario, so inducing instability. A scenario can also occur where some players are requesting chunks with lower bitrates than the other players. This can occur as some players observe a throughput lower than the fair share, while others observe a throughput that is more than the fair share. This means that some players overestimate its fair share. When some players overestimate their fair share, it can be that the system of players converge to a stable equilibrium, but unfair. This occurs as the players with the larger fair share estimates request higher bitrate video levels. Even in the case where two players estimate their fair share correctly, bandwidth underutilization can still be prevalent. This occurs as both players request the same lower video bitrate level, which causes underutilization, even though stability and fairness still exist. In reality, several other factors can play an important role in the appearance and extent of instability, unfairness and underutilization, such as the exact player adaptation algorithm, TCP dynamics, bandwidth fluctuations, and the variability of the video encoding rate [1]. We group these problems into three categories: The first relates to the stability of the players in terms of requested bitrates and video quality. The second is the unfairness among competing players. The third is the potential bandwidth underutilization when multiple adaptive players compete.

III. LITERATURE REVIEW

The literature review is divided into the three most popular methods for video adaptive streaming, proxy-based, server-based and client-based. It has been shown that today's adaptive streaming techniques underperform when multiple clients consume video at the same time, due to fairness issues among clients. Concretely, this means that different clients negatively influence each other as they compete for shared network resources. FINEAS (Fair In-Network Enhanced Adaptive Streaming) is proposed [31], which is capable of increasing clients' Quality of Experience (QoE) and achieving fairness in a multi-client setting. A key element of their approach is an in-network system of coordination proxies in charge of facilitating fair resource sharing among clients. They claim that fairness is achieved without explicit communication among clients [31]. In addition, viewers using HTTP Adaptive Streaming (HAS) without sufficient bandwidth undergo frequent quality switches that hinder their watching experience. This situation, known as instability, is produced when HAS players are unable to accurately estimate the available bandwidth. Moreover, when several players stream over a bottleneck link, their individual adaptation techniques may result in an unfair share of the channel. These are two detrimental issues in HAS technology, which is otherwise very attractive. The authors [20] describe an implementation in the form of an HTTP proxy server and show that both stability and fairness are strongly improved. In [6] several network-assisted streaming approaches which rely on active cooperation between video streaming applications and the network are explored. They use a Video Control Plane which enforces

Volume: 10 Issue: 01 Pages: 3704-3712 (2018) ISSN: 0975-0290

Video Quality Fairness among concurrent video flows generated by heterogeneous client devices. A max-min fairness optimization problem is solved at run-time. They compare two approaches to actuate the optimal solution in an SDN network: the first one allocating network bandwidth slices to video flows, the second one guiding video players in the video bitrate selection.

In [36] the bandwidth estimate generated at the server is used for server-side adaptive bit encoding of digital media streams. The server application measures the network bandwidth available to the individual client for TCP/IP downloads of media and accordingly adjusts stream bit rate and composition to allow the client to retrieve the media stream with sufficient time margin to minimize the occurrence of underflow of client playback buffers. The root cause of the instability problem is that, in Steady-State, a player goes through an ON-OFF activity pattern in which it overestimates the available bandwidth [2]. They propose a server-based traffic shaping procedure that can considerably lower such oscillations. Their procedure is only triggered when oscillations are identified, and so the shaping rate is dynamically adjusted. This ensures that the player receives the highest available video profile without being unstable. Using HTTP for video streaming significantly increases the request overhead due to the segmentation of the video content into HTTP resources [43]. This overhead becomes even more substantial when non-multiplexed video and audio segments are deployed. The authors investigate the request overhead problem by employing the server push technology in the new HTTP 2.0 protocol. They develop a set of push strategies that actively deliver video and audio content from the HTTP server without requiring a request for each individual segment.

Chunk scheduling with stateless bitrate selection causes feedback loops, bad bandwidth estimation, bitrate switches and unfair bitrate choices [18]. This paper, which portrays the FESTIVE control algorithm, confirms that numerous problems occur when multiple bitrate-adaptive players (adaptation over HTTP) share a bottleneck link [46]. It uncovers the fact that the feedback signals the player receives is not a true reflection of the network state because of overlaying the adaptation logic over several layers. HTTP-based video delivery issues are elucidated: (1) the granularity of the control decisions, (2) the timescales of adaptation, (3) the nature of feedback from the network and (4) the interactions with other independent control loops in lower layers of the networking stack. FESTIVE uses an abstract player state to analyze commercial players: (1) schedule a video chunk for download, (2) select bitrate for chunk, and (3) estimate bandwidth. It identifies root causes of undesirable interactions with abstract state player framework and saw the need to guide the tradeoffs between stability, fairness and efficiency. As a result, the authors created a robust video adaptation algorithm, which tried to achieve: (1) Fairness – equal allocation of network resources, (2) Efficiency – get highest bitrates for maximum user experience, and (3) Stability – avoid needless bitrate switches. The eventual contributions were a family of adaptation algorithms using the following approaches: (1) Randomized chunk scheduling: to avoid sync biases in network state sampling, (2) Stateful bitrate selection: to compensate between biased bitrate and estimated bandwidth interaction, (3) Delayed update: to account for stability and efficiency tradeoff, and (4) Bandwidth estimator: to increase robustness to outliers.

The authors in [22], who proposed the PANDA algorithm, noted that since TCP throughput observed by a client would indicate the available network bandwidth, it could be used as a reliable reference for video bitrate selection. However, this is no longer true when HTTP Adaptive Streaming (HAS) [4] becomes a substantial fraction of the total network traffic or when multiple HAS clients compete at a network bottleneck. It was observed that the discrete nature of the video bitrates results in difficulty for a client to correctly perceive its fair-share bandwidth. Hence, this fundamental limitation would lead to video bitrate oscillation and other undesirable behaviors that negatively impact the video viewing experience. They offered a design at the application layer using a “probe and adapt” principle for video bitrate adaptation (where “probe” refers to trial increment of the data rate, instead of sending auxiliary piggybacking traffic), which is akin, but also orthogonal to the transport-layer TCP congestion control. The authors illustrate a four-step state for an HAS rate adaptation algorithm: (1) Estimating: the algorithm starts by estimating the network bandwidth that can legitimately be used, (2) Smoothing: is then noise-filtered to yield the smoothed version, with the aim of removing outliers, (3) Quantizing: the continuous is then mapped to the discrete video bitrate, possibly with the help of side information such as client buffer size [42], [12], [25] etc., and (4) Scheduling: the algorithm selects the target interval until the next download request. The advantages of PANDA are as follows. Firstly, as the bandwidth estimation by probing is quite accurate, one does not need to apply strong smoothing. Secondly, since after a bandwidth drop, the video bitrate reduction is made proportional to the TCP throughput reduction, PANDA is very sensitive to bandwidth drops.

ELASTIC [8] proposes an approach that uses one controller to throttles the video level (t). This drives the playout buffer length (t) to a set-point q_T, which eliminates the ON-OFF traffic pattern. The player is always in ON phase unless (t) is the highest level and $q > Q_{max}$ ($>q_T$). The basic concept is based on the playout buffer state, design a feedback control system that computes l(t) to steer q(t) to a threshold q_T. The received rate r(t), is considered as a (measurable) disturbance since it cannot be manipulated. ELASTIC provides a received video rate that oscillates around the fair share, with an increased number of video level switches. However, the main result involved long-lived TCP flows [44], where experimental evaluation showed that ELASTIC is able to get the fair share when competing with TCP long-lived flows.

To the authors review of existing literature there is no known findings of adaptive streaming players with distributed client-to-client communications. We propose, implement and test two algorithms for players using distributed client-to-client communication. It primarily

Volume: 10 Issue: 01 Pages: 3704-3712 (2018) ISSN: 0975-0290

aims to obtain better fair share in environments where adaptive video player clients are able to communicate with each other, for example, in a home or company's local area network (LAN).

IV. VARYING BANDWIDTH

In practice, available network resource for media streaming can change over time due to fluctuating link capacity in wireless networks, [34] or influence of other traffic. In an environment with stable capacity, past observations yield good estimates of future capacity. But, if capacity is varying widely, estimating future capacity is much harder, [14]. In general, large buffer sizes compensates network bandwidth variations (time-varying bandwidth). Ample buffering of too many segments guarantees a smooth video rate, [38].

The research community utilizes the term "available bandwidth" in varying context. We adopt available bandwidth as, [33], does. In a network path each link j has a certain capacity or nominal bandwidth, C_j. The network interfaces in the nodes at each end of the link determine this value. The nominal bandwidth typically does not vary, \cite{ekelin2006real}. Usually, varying bandwidth happens within short time-scales. It is based on the link load, or cross traffic $Y_j=Y_j(t, \lambda)$, where λ is the time resolution describing traffic fluctuations. The cross-traffic rate is given by Equation:

$$Y_j = \frac{1}{\lambda} A_j(t-\lambda, t)$$

where $A_j(t- \lambda, t)$ is the number of bits over link j during a time interval λ.

The time-varying bandwidth $B_j=B_j(t, \lambda)$ of the link j is given by Equation:

$$B_j = C_j - Y_j$$

The bottleneck link is one of the links along the path that has the smallest available bandwidth value. It determines the available bandwidth of the path. The available bandwidth is the smallest increase in traffic load from sender to receiver at time t, which causes congestion at some hop on the network path, [16].

Overlapping ON-OFF traffic occurs when players compete for bandwidth at a bottleneck link. This results in poor user-QoE. In addition, (when the bandwidth becomes too high or low in consecutive time intervals) causes user-QoE too quality to suffer even further.

V. EXPERIMENTAL SETUP

The Controller code was written in python. Each player used a port to send broadcast messages periodically to the other players when in BEGGAR state. In addition, all players had a receiver thread running which listens for incoming messages. TAPAS [9], an open-source Tool for rApid Prototyping of Adaptive Streaming control algorithms. TAPAS is a flexible and extensible video streaming client written in python that allows researchers to easily design and carry out experimental performance evaluations of adaptive streaming controllers without needing to write the code to download video segments, parse manifest files, and decode the video. TAPAS have been designed to minimize the CPU and memory footprint so that experiments involving a large number of concurrent video flows can be carried out. The player logs experimental data results. The TAPAS player communicates with the video server in the form of a GET request. The Controller has access to a shared table which contains INFO data. The INFO packet is small and thus imposes very low overhead to the BEGGAR protocol. This data is used to calculate Fair share bandwidth and to then request or reduce the player's bitrate request to the server.

A virtual network is setup on the same host machine creating a custom emulation framework. Our setup consists of client players, video servers, and a bottleneck link. The server resides on a Windows 10 machine. All experiments are performed on a Windows 10 client with an Intel(R) Core(TM)i7-5500U CPU 2.40GHz processor, 16.00 GB physical memory, and an Intel(R) HD Graphics processor. It serves video data to the client(s) who are on a Ubuntu operating system hosted on VMware. The virtual machine is allocated 12GB of physical memory. TAPAS is installed on Ubuntu 15.04 Linux. The TAPAS Adaptive Video Controller client makes different video segment bitrate level requests to the Apache server. TAPAS allow multiple instances of the player to be created enabling multi-client scenarios. This work involves the interaction between adaptive streaming algorithm at the controller and TAPAS player (cf. Figure 6). All traffic between clients and servers go through the bottleneck, which uses VMware settings which allow bandwidth limits to be set during the experiment. TAPAS support both the HTTP Live Streaming (HLS) and Dynamic Adaptive Streaming over HTTP (DASH) format. Algorithms that uses the BEGGAR protocol was tested and shown to work on both MPEG-DASH [39], and Apple HTTP Live Streaming (HLS) [35]. This makes it useful for video on demand (VOD) [30] and live streaming [24], for example, real-time video chats. However, the MPEG-DASH standard is used for testing in this research paper, because it makes the experiments more comparable to the ones in the research literature, for example, [8]. The ten-minute-long MPEG-DASH video sequence "Elephant's Dream"[1] is encoded at twenty different bitrates, between 46 Kbps to 4200Kbps and five different resolutions, between 320x240 to 1920x1080, is used to run the experiments (cf. Table II). The video is encoded at 24 frames per second (fps) using the AVC1 codec [15]. Fragment duration of 2s is used and is recorded in the mpd playlist accordingly. All the DASH files (.m4s fragments and .mpd playlists) are placed on the Apache server. We implemented three client-side algorithms in the TAPAS controller. The conventional approach is present by default and is used as a baseline in which to compare against other algorithms. TAPAS is lightweight in built, thus allowing the same receiving host to run a large number of separate video player instances at the same time at different command line interfaces. Thus, it allows the

Volume: 10 Issue: 01 Pages: 3704-3712 (2018) ISSN: 0975-0290

multi-client scenarios which are essential to the work in this paper.

The experiment considers a bottleneck link [46] with two total video connections. The available bandwidth is set to b = 10Mbps. QoE metrics are described as follows:

i. The unfairness metric (for two players) is the average of the absolute bitrate difference between the corresponding chunks requested by each player (cf. Equation 5, where p1 and p2 are player 1 and player 2, respectfully). The bitrate is the number of bits required to encode one second of playback.

$$Unfairness = Average(\sum_{i=0}^{n-1} |r_{i,p1} - r_{i,p2)}|) \quad (5)$$

ii. The utilization metric is defined as the aggregate throughput during an experiment divided by the available bandwidth in that experiment (cf. Equation 6, where tp_i is the throughput at time i and bw is the experimental available bandwidth).

$$Utilization = \frac{\sum_{i=0}^{n-1} tp_i}{bw} \quad (6)$$

In the experiment (E2) the instability, inefficiency, and unfairness (different formulae used for the multi-player scenario) metrics, and re-buffering ratios is used to compare the performances of the considered algorithms.

i. Instability: The instability for player i at time t is given in Equation 7, where $w(d) = k - d$ is a weight function that puts more weight on more recent samples. k is selected as 20 seconds.

$$Instability = \frac{\sum_{d=0}^{k-1} |r_{i,t-d} - r_{i,t-d-1}| * w(d)}{\sum_{d=0}^{k-1} r_{i,t-d} * w(d)} \quad (7)$$

ii. Inefficiency: The inefficiency at time t is given in Equation 8. Consider N players sharing a bottleneck link with bandwidth, w, with each player x, playing a bit rate, $b_{x,t}$, at time t. A value close to zero implies that the players in aggregate are using as high an average bitrate as possible to improve user experience.

$$Inefficiency = \left| \frac{\sum_x b_{x,t-w}}{w} \right| \quad (8)$$

iii. Unfairness: Let $JainFair_t$ be the Jain fairness index (cf. Equation 10) calculated on the average received rates [8], r_i, (cf. Equation 9) at time t over all players. The unfairness at time t is defined as $\sqrt{1 - JainFair_t}$. A lower value implies a fairer allocation.

$$r_i = \frac{downloaded\ bytes}{time\ interval} \quad (9)$$

$$JFI = \frac{(\sum_{i=1}^{n} r_i)^2}{n \sum_{i=1}^{n} r_i^2} \quad (10)$$

iv. Re-buffering ratio: is the ratio of the time spent in re-buffering and the total playtime of the stream Equation 11.

$$Re-buffering\ ratio = \frac{total\ re-buffering\ time}{experiment\ duration} \quad (11)$$

We utilize quality, re-buffering and instability in our results which is presented in the upcoming section.

VI. RESULTS

We first present the level curves which represent the incoming bitrates of players, see Figures 1, 2, 3 and 4. We observe PANDA with the best level curves among the competing two players. FESTIVE does the second best with ELASTIC the third and the Conventional doing the worst. PANDA's probe mechanism is able to detect bandwidth changes and the adapt mechanism helps the player to cope well with time-varying bandwidth environments.

This result is also shown on Figures 5, 6 and 7. PANDA is a heuristic which outperforms the other approaches in most cases.

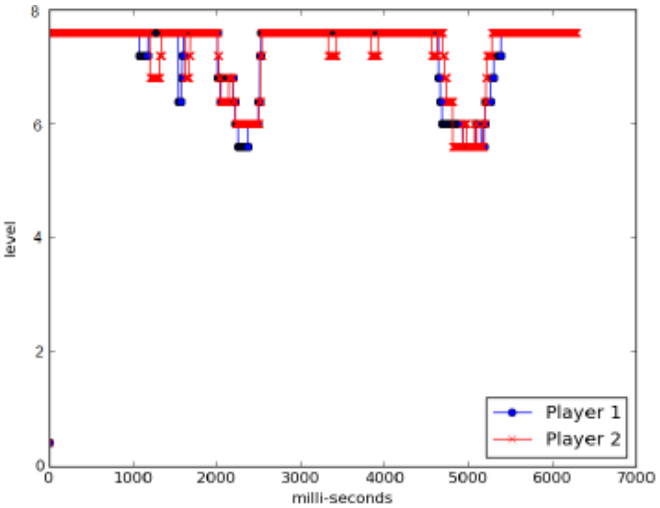

Figure 1: PANDA LEVEL CURVE: BANDWIDTH VARIATIONS

Volume: 10 Issue: 01 Pages: 3704-3712 (2018) ISSN: 0975-0290

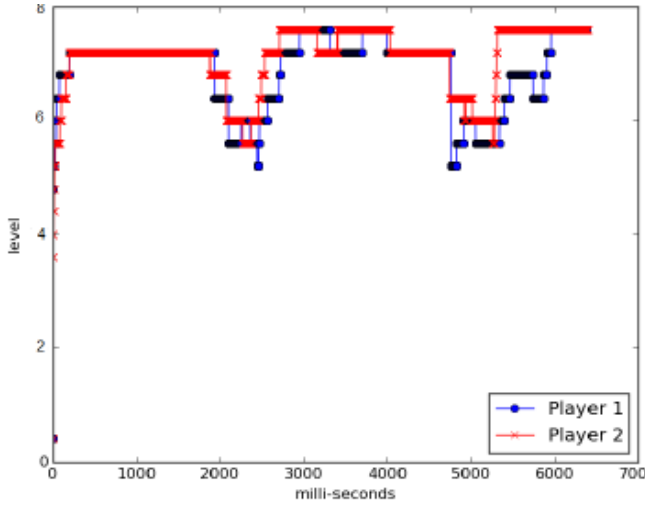

Figure 2: FESTIVE LEVEL CURVE: BANDWIDTH VARIATIONS

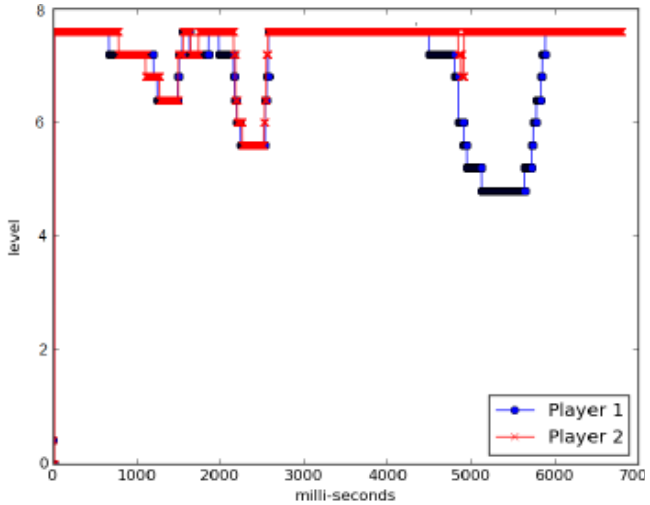

Figure 3: ELASTIC LEVEL CURVE: BANDWIDTH VARIATIONS

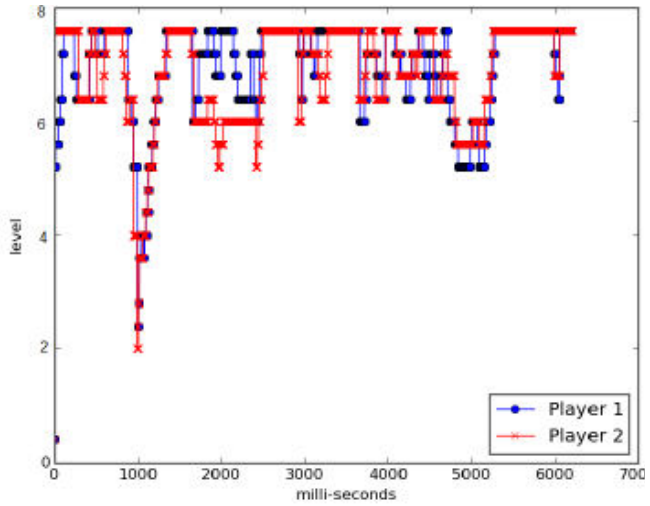

Figure 4: CONVENTIONAL LEVEL CURVE: BANDWIDTH VARIATIONS

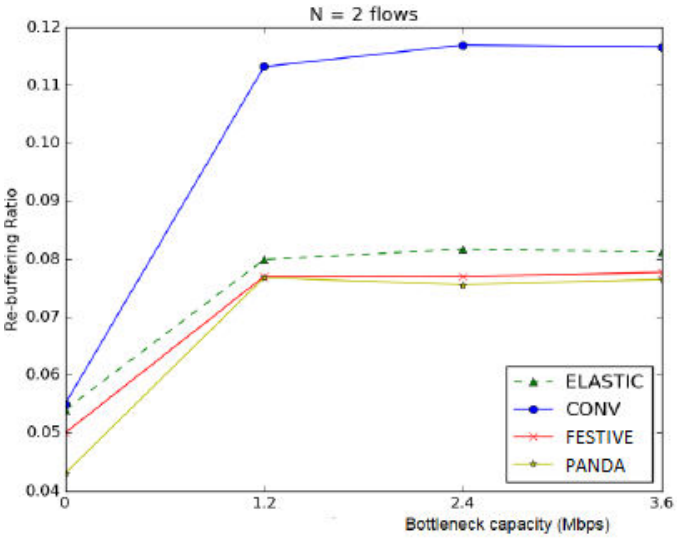

Figure 5: RE-BUFFERING RATIO: BANDWIDTH VARIATIONS - 1.2 MBPS, 2.4 MBPS AND 3.6 MBPS.

VII. CONCLUSION

Competition among adaptive video streaming players severely diminishes user-QoE. When players compete at a bottleneck link many do not obtain adequate resources. This imbalance eventually causes ill effects such as screen flickering and video stalling. There have been many attempts in recent years to overcome some of these problems. However, added to the competition at the bottleneck link there is also the possibility of varying network bandwidth which can make the situation even worse. This work focuses on such a situation. It evaluates current heuristic adaptive video players at a bottleneck link with time-varying bandwidth conditions. Experimental setup includes the TAPAS player and emulated network conditions. The results show PANDA outperforms FESTIVE, ELASTIC and the Conventional players.

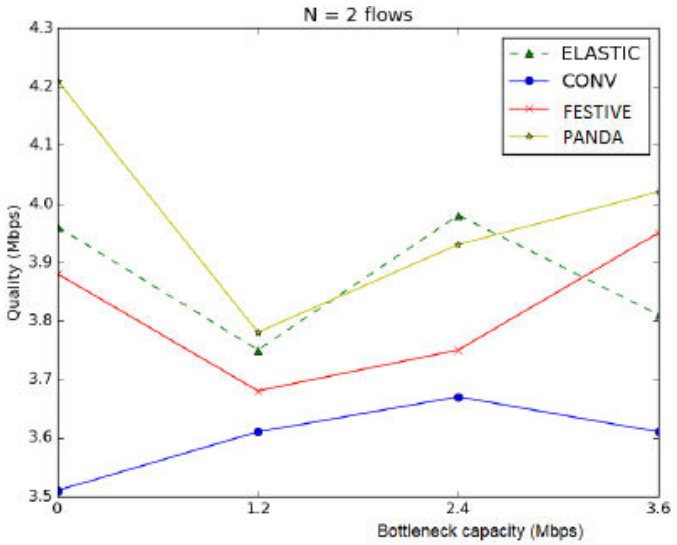

Figure 6: AVERAGE QUALITY: BANDWIDTH VARIATIONS - 1.2 MBPS, 2.4 MBPS AND 3.6 MBPS.

Volume: 10 Issue: 01 Pages: 3704-3712 (2018) ISSN: 0975-0290

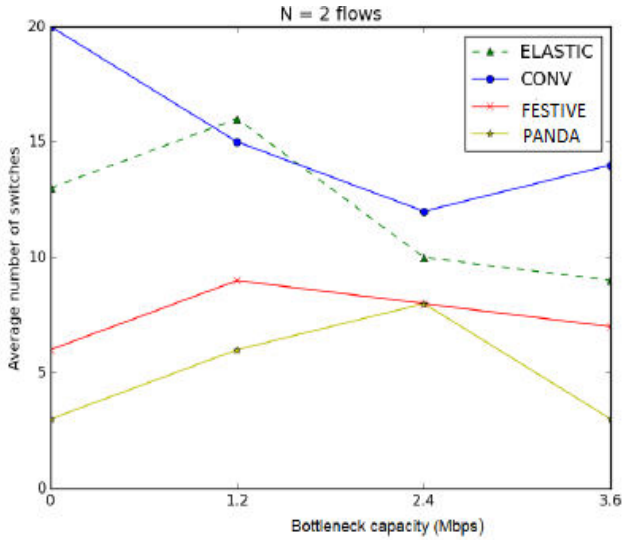

Figure 7: AVERAGE NUMBER OF SWITCH EVENTS: BANDWIDTH VARIATIONS - 1.2 MBPS, 2.4 MBPS AND 3.6 MBPS.

REFERENCES

[1] Akhshabi, Saamer, Lakshmi Anantakrishnan, Ali C. Begen, and Constantine Dovrolis. "What happens when HTTP adaptive streaming players compete for bandwidth?." In Proceedings of the 22nd international workshop on Network and Operating System Support for Digital Audio and Video, pp. 9-14. ACM, 2012.

[2] Akhshabi, Saamer, Lakshmi Anantakrishnan, Constantine Dovrolis, and Ali C. Begen. "Server-based traffic shaping for stabilizing oscillating adaptive streaming players." In Proceeding of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video, pp. 19-24. ACM, 2013.

[3] Akhshabi, Saamer, Sethumadhavan Narayanaswamy, Ali C. Begen, and Constantine Dovrolis. "An experimental evaluation of rate-adaptive video players over HTTP." Signal Processing: Image Communication 27, no. 4 (2012): 271-287.

[4] Bouten, Niels, Steven Latré, Jeroen Famaey, Werner Van Leekwijck, and Filip De Turck. "In-network quality optimization for adaptive video streaming services." IEEE Transactions on Multimedia 16, no. 8 (2014): 2281-2293.

[5] Chen, Liang, Yipeng Zhou, and Dah Ming Chiu. "Smart streaming for online video services." IEEE Transactions on Multimedia 17, no. 4 (2015): 485-497.

[6] Cofano, G., L. De Cicco, T. Zinner, A. Nguyen-Ngoc, P. Tran-Gia, and S. Mascolo. "Design and experimental evaluation of network-assisted strategies for HTTP adaptive streaming." In Proceedings of the 7th International Conference on Multimedia Systems, p. 3. ACM, 2016.

[7] De Cicco, Luca, and Saverio Mascolo. "An adaptive video streaming control system: Stateing, validation, and performance evaluation." IEEE/ACM Transactions on Networking (TON) 22, no. 2 (2014): 526-539.

[8] De Cicco, Luca, Vito Caldaralo, Vittorio Palmisano, and Saverio Mascolo. "Elastic: a client-side controller for dynamic adaptive streaming over http (dash)." In 2013 20th International Packet Video Workshop, pp. 1-8. IEEE, 2013.

[9] De Cicco, Luca, Vito Caldaralo, Vittorio Palmisano, and Saverio Mascolo. "TAPAS: a Tool for rApid Prototyping of Adaptive Streaming algorithms." In Proceedings of the 2014 Workshop on Design, Quality and Deployment of Adaptive Video Streaming, pp. 1-6. ACM, 2014.

[10] Dobrian, Florin, Vyas Sekar, Asad Awan, Ion Stoica, Dilip Joseph, Aditya Ganjam, Jibin Zhan, and Hui Zhang. "Understanding the impact of video quality on user engagement." In ACM SIGCOMM Computer Communication Review, vol. 41, no. 4, pp. 362-373. ACM, 2011.

[11] Ginevičius, Romualdas. "Normalization of quantities of various dimensions."Journal of business economics and management 9, no. 1 (2008): 79-86.

[12] He, Jian, Zheng Xue, Di Wu, Dapeng Oliver Wu, and Yonggang Wen. "CBM: online strategies on cost-aware buffer management for mobile video streaming." IEEE Transactions on Multimedia 16, no. 1 (2014): 242-252.

[13] Huang, Te-Yuan, Nikhil Handigol, Brandon Heller, Nick McKeown, and Ramesh Johari. "Confused, timid, and unstable: picking a video streaming rate is hard." In Proceedings of the 2012 ACM conference on Internet measurement conference, pp. 225-238. ACM, 2012.

[14] Huang, Te-Yuan, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. "A buffer-based approach to rate adaptation: Evidence from a large video streaming service." ACM SIGCOMM Computer Communication Review 44, no. 4 (2015): 187-198.

[15] Irondi, Iheanyi, Qi Wang, and Christos Grecos. "Empirical evaluation of H. 265/HEVC-based dynamic adaptive video streaming over HTTP (HEVC-DASH)." In SPIE Photonics Europe, pp. 91390L-91390L. International Society for Optics and Photonics, 2014.

[16] Jacobson, Van. "Congestion avoidance and control." In ACM SIGCOMM computer communication review, vol. 18, no. 4, pp. 314-329. ACM, 1988.

[17] Jarnikov, Dmitri, and Tanır Özçelebi. "Client intelligence for adaptive streaming solutions." Signal Processing: Image Communication 26, no. 7 (2011): 378-389.

[18] Jiang, Junchen, Vyas Sekar, and Hui Zhang. "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive." InProceedings of the 8th international conference on Emerging networking experiments and technologies, pp. 97-108. ACM, 2012.

[19] Juluri, Parikshit, Venkatesh Tamarapalli, and Deep Medhi. "Measurement of Quality of Experience of Video-on-Demand Services: A Survey." IEEE

Volume: 10 Issue: 01 Pages: 3704-3712 (2018) ISSN: 0975-0290

Communications Surveys & Tutorials 18, no. 1 (2016): 401-418.

[20] Kleinrouweler, Jan Willem, Sergio Cabrero, Rob van der Mei, and Pablo Cesar. "Modeling stability and bitrate of network-assisted HTTP adaptive streaming players." In Teletraffic Congress (ITC 27), 2015 27th International, pp. 177-184. IEEE, 2015.

[21] Lederer, Stefan, Christopher Müller, and Christian Timmerer. "Dynamic adaptive streaming over HTTP dataset." In Proceedings of the 3rd Multimedia Systems Conference, pp. 89-94. ACM, 2012.

[22] Li, Zhi, Xiaoqing Zhu, Joshua Gahm, Rong Pan, Hao Hu, Ali C. Begen, and David Oran. "Probe and adapt: Rate adaptation for http video streaming at scale." IEEE Journal on Selected Areas in Communications 32, no. 4 (2014): 719-733.

[23] Liu, Chenghao, Imed Bouazizi, Miska M. Hannuksela, and Moncef Gabbouj. "Rate adaptation for dynamic adaptive streaming over HTTP in content distribution network." Signal Processing: Image Communication 27, no. 4 (2012): 288-311.

[24] Magharei, Nazanin, Reza Rejaie, Ivica Rimac, Volker Hilt, and Markus Hofmann. "ISP-friendly live P2P streaming." IEEE/ACM Transactions on Networking 22, no. 1 (2014): 244-256.

[25] Mansy, Ahmed, Bill Ver Steeg, and Mostafa Ammar. "Sabre: A client based technique for mitigating the buffer bloat effect of adaptive video flows." InProceedings of the 4th ACM Multimedia Systems Conference, pp. 214-225. ACM, 2013.

[26] Miller, Konstantin, Dilip Bethanabhotla, Giuseppe Caire, and Adam Wolisz. "A control-theoretic approach to adaptive video streaming in dense wireless networks." IEEE Transactions on Multimedia 17, no. 8 (2015): 1309-1322.

[27] Miller, Konstantin, Emanuele Quacchio, Gianluca Gennari, and Adam Wolisz. "Adaptation algorithm for adaptive streaming over HTTP." In 2012 19th International Packet Video Workshop (PV), pp. 173-178. IEEE, 2012.

[28] Mueller, Christopher, Stefan Lederer, and Christian Timmerer. "A proxy effect analyis and fair adatpation algorithm for multiple competing dynamic adaptive streaming over HTTP clients." In Visual Communications and Image Processing (VCIP), 2012 IEEE, pp. 1-6. IEEE, 2012.

[29] Mueller, Christopher, Stefan Lederer, and Christian Timmerer. "A proxy effect analyis and fair adatpation algorithm for multiple competing dynamic adaptive streaming over HTTP clients." In Visual Communications and Image Processing (VCIP), 2012 IEEE, pp. 1-6. IEEE, 2012.

[30] Nikmanzar, Sepideh, Akbar Ghaffarpour Rahbar, and Amin Ebrahimzadeh. "On-Demand Video Streaming Schemes Over Shared-WDM-PONs." IEEE Transactions on Circuits and Systems for Video Technology 23, no. 9 (2013): 1577-1588.

[31] Petrangeli, Stefano, Jeroen Famaey, Maxim Claeys, Steven Latré, and Filip De Turck. "QoE-driven rate adaptation heuristic for fair adaptive video streaming." ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 12, no. 2 (2016): 28.

[32] Prasad, Ravi, Constantinos Dovrolis, Margaret Murray, and K. C. Claffy. "Bandwidth estimation: metrics, measurement techniques, and tools." IEEE network 17, no. 6 (2003): 27-35.

[33] Prasad, Ravi, Constantinos Dovrolis, Margaret Murray, and K. C. Claffy. "Bandwidth estimation: metrics, measurement techniques, and tools." IEEE network 17, no. 6 (2003): 27-35.

[34] Raychaudhuri, Dipankar, and Narayan B. Mandayam. "Frontiers of wireless and mobile communications." Proceedings of the IEEE 100, no. 4 (2012): 824-840.

[35] Robinson, David C., Yves Jutras, and Viorel Craciun. "Subjective video quality assessment of HTTP adaptive streaming technologies." Bell Labs Technical Journal 16, no. 4 (2012): 5-23.

[36] Schmidt, Mark S., Praveen N. Moorthy, and Baozhou Li. "Server-side adaptive bit rate control for dlna http streaming clients." U.S. Patent Application 14/991,091, filed January 8, 2016.

[37] Seufert, Michael, Sebastian Egger, Martin Slanina, Thomas Zinner, Tobias Hoßfeld, and Phuoc Tran-Gia. "A survey on quality of experience of http adaptive streaming." IEEE Communications Surveys & Tutorials 17, no. 1 (2015): 469-492.

[38] Spiteri, Kevin, Rahul Urgaonkar, and Ramesh K. Sitaraman. "BOLA: Near-optimal bitrate adaptation for online videos." In INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, IEEE, pp. 1-9. IEEE, 2016.

[39] Stockhammer, Thomas. "Dynamic adaptive streaming over HTTP--: standards and design principles." In Proceedings of the second annual ACM conference on Multimedia systems, pp. 133-144. ACM, 2011.

[40] Su, Guan-Ming, Xiao Su, Yan Bai, Mea Wang, Athanasios V. Vasilakos, and Haohong Wang. "QoE in video streaming over wireless networks: perspectives and research challenges." Wireless Networks (2015): 1-23.

[41] Thang, Truong Cong, Quang-Dung Ho, Jung Won Kang, and Anh T. Pham. "Adaptive streaming of audiovisual content using MPEG DASH." IEEE Transactions on Consumer Electronics 58, no. 1 (2012): 78-85.

[42] Wamser, Florian, David Hock, Michael Seufert, Barbara Staehle, Rastin Pries, and Phuoc Tran-Gia. "Using buffered playtime for QoE-oriented resource management of YouTube video streaming." Transactions on Emerging Telecommunications Technologies 24, no. 3 (2013): 288-302.

[43] Wei, Sheng, and Viswanathan Swaminathan. "Cost effective video streaming using server push over HTTP 2.0." In Multimedia Signal Processing (MMSP), 2014 IEEE 16th International Workshop on, pp. 1-5. IEEE, 2014.

[44] Wichtlhuber, Matthias, Robert Reinecke, and David Hausheer. "An SDN-based CDN/ISP collaboration

Volume: 10 Issue: 01 Pages: 3704-3712 (2018) ISSN: 0975-0290

architecture for managing high-volume flows."IEEE Transactions on Network and Service Management 12, no. 1 (2015): 48-60.

[45] Wu, Jiyan, Bo Cheng, Chau Yuen, Ngai-Man Cheung, and Junliang Chen. "Trading delay for distortion in one-way video communication over the internet." IEEE Transactions on Circuits and Systems for Video Technology26, no. 4 (2016): 711-723.

[46] Yin, Xiaoqi, Vyas Sekar, and Bruno Sinopoli. "Toward a principled framework to design dynamic adaptive streaming algorithms over http." In Proceedings of the 13th ACM Workshop on Hot Topics in Networks, p. 9. ACM, 2014.

[47] Yu, Hongliang, Dongdong Zheng, Ben Y. Zhao, and Weimin Zheng. "Understanding user behavior in large-scale video-on-demand systems." InACM SIGOPS Operating Systems Review, vol. 40, no. 4, pp. 333-344, ACM, 2006.

[48] Zhou, Chao, Chia-Wen Lin, Xinggong Zhang, and Zongming Guo. "A control-theoretic approach to rate adaption for DASH over multiple content Distribution servers. "IEEE Transactions on Circuits and Systems for Video Technology 24, no. 4 (2014): 681-694.

AUTHORS BIOGRAPHY

Koffka Khan received the M.Sc., and M.Phil. degrees from the University of the West Indies. He is currently a PhD student and has up-to-date, published numerous papers in journals & proceedings of international repute. His research areas are computational intelligence, routing protocols, wireless communications, information security and adaptive streaming controllers.

Wayne Goodridge is a Lecturer in the Department of Computing and Information Technology, The University of the West Indies, St. Augustine. He did is PhD at Dalhousie University and his research interest includes computer communications and security.